Table of Contents

Summary i
Acknowledgements ii
Chapter 1 Introduction 1
1.1 Statement of the Problem 1
1.2 Aims and Objectives 1
1.3 Minimum Requirements 1
1.4 Project Schedule 2
Chapter 2 Preparation 3
2.1 An Overview of Athlete Training 3
2.2 Technologies 3
2.2.1 Database Technologies 3
2.2.1.1 MySQL 4
2.2.1.2 Postgres 4
2.2.1.3 Conclusion 4
2.2.2 Server Side Scripting Technologies 4
2.2.2.1 Perl 5
2.2.2.2 Python 5
2.2.2.3 PHP 5
2.2.2.4 The Java Family 5
2.2.2.5 Conclusion 6
2.2.3 Web Design 6
2.2.4 User Tracking 6
2.2.4.1 Sessions 6
2.2.4.2 Cookies 7
2.2.4.3 Conclusion 7
2.3 Past Projects 7
2.4 Methodologies 7
2.4.1 The Waterfall Model 7
2.4.2 The Spiral Model 8
2.4.3 The 'V' Model 8
2.4.4 Rapid Application Development (RAD) 8
2.4.5 Conclusion 9
2.5 Human Computer Interaction (HCI) 9
2.6 Functional Requirements 9

Chapter 3 Design 10
3.1 Overview 10
3.2 Structure of the System 10
3.2.1 Database 11
3.2.2 User Interface 14
3.2.3 PHP Scripts 15
3.2.4 User Tracking/Cookies 15
Chapter 4 Implementation 16
4.1 Database 16
4.2 HTML/User Interface 16
4.3 PHP 16
4.3.1 Retrieving Data 17
4.3.2 Inserting Data 17
4.3.3 Page by Page 18
4.4 Security 19
4.5 Implementation Problems 20
4.5.1 User Authentication 20
4.5.2 User Tracking 20
4.5.3 Conclusion 21
Chapter 5 Testing 22
5.1 System Testing 22
5.2 HCI Testing 24
5.3 Browser Compatibility Testing 25
Chapter 6 Evaluation 26
6.1 Evaluation Criteria 26
6.1.1 Minimum Requirements 26
6.1.2 Functional Requirements 27
6.2 Methodology 27
6.3 Project Management 27
6.4 Heuristic Evaluation 28
6.5 Evaluation Summary 28
6.6 Future Enhancements 28
6.6.1 Functional Enhancements 28
6.6.2 Security Enhancements 29
Bibliography 30
Appendix A Personal Reflection 32
Appendix B Usability Questionnaire 34

Appendix C User Manual 35

Summary

Athletic competition has existed ever since man could walk. Over the tens of thousands of years since our species evolved, it has become the most important thing in many peoples lives. One need only look at the Kop end when Liverpool score, or the crowd when England lift the ashes to see that.

Track and field athletics, although not as popular as its more glamorous rivals, is perhaps the oldest form of athletic competition in the world, and yet computer based systems to help them out with training are few and far between.

This project looked at the needs of athletes, and a Distributed system was created in order to facilitate their training.

Acknowledgements

Where to start?

First of all thanks to my assessor Jie Xu, for helping me out in all aspects of the project.

Thanks to Bucky, Cat and Dave for having to do projects too, at least I wasn't alone in my suffering.

Thanks to Liv, Jess, Scouse, Scott, Rob, Frank, Frup, Roge, Simon, Hazy, OJ and anyone else who has given me support throughout this project.

And thanks to people like Ste Gray, Guile Lopez, and the scores of people who have helped me out with work throughout my time here. Without you it's probable I would have been kicked out after 1st year.

Finally, thanks to the University, and City of Leeds, for the most amazing 4 years of my life. I wish I could do it all again.

1.1 The Problem

As an athlete, either professional or amateur, training for ones events is a gruelling physical experience in which a millisecond or centimetre can make all the difference. They must treat their bodies in the same way as a high performance engine, treating it well, giving it the proper fuel, and making sure output does not drop. To do this they must train many hours a day, meaning they have little time for the administration side of their training; tasks such as recording their successes and failures in training and keeping track of any injuries they may sustain. It would also be desirable for the athletes to be able to generate reports on their progress.

Coaches also have time management issues. They often find themselves stretched between many different athletes in different parts of the country, meaning that valuable training time is lost through travel, time which could be better used. A distributed system, accessable from any place at any time, and which is capable of storing training schedules and training times, as well as other useful information would appear to be the ideal solution.

1.2 Aims and Objectives

The aim of this project is to create a fully working prototype of a system which athletes can use to log records of their training and injuries, and generate reports of sed information. The system must also support “coaching at a distance” allowing coaches to log in, view their athletes progress, and send them messages.

1.3 Minimum Requirements

The Minimum Requirements are:

1. Understand athlete’s training systems & gather basic user requirements for the system
2. A Database capable of storing all required information
3. A Web page with a form to enter new training regimes and times
4. Some basic CGI to take the form's information and insert it into database tables
5. Some basic CGI capable of querying the database, and displaying returned rows in a suitable format.

1.4 Schedule

Date	Objective
22nd Nov – 26th Nov	Background reading on Dist Syst. Begin Mid Project Report(MPR)
29th Nov – 3rd Dec	Finish off Distributed Systems reading Begin basic design elements – web and database Submit 1st draft of MPR
6th Dec – 10th Dec	Decision on technologies to be used Submit final MPR
13th Dec – 22nd Dec	Implement database system Create web pages Further reading on athlete training
22nd Dec – 10th Jan	Exam revision + more reading. Create complete list of functionalities
10th Jan – 21st Jan	Exam period 3 free days at end of exams to catch up on anything I'm behind in
24th Jan – 29th Jan	Web and database systems finished Begin design of CGI/java
1st Feb – 11th Feb	Completed CGI/java design Begin CGI/java implementation Aim to have fully working system prototype by 11th.
14th Feb – 18th Feb	Write evaluation criteria Bug test system, have version 1.0 by 18th
22nd Feb – 18th Mar	Evaluate system Write up evaluation Write users manual
21st Mar – 4th Apr	Write conclusions Put together final report Submit 1st draft of final report
4th Apr – 18th Apr	Continual re-drafting of final report Submit Final Report

Chapter 2: Preparation

2.1 An Overview of Distributed Systems

2.1.1 Client/Server Model

The term client/server denotes a software system where processing takes place between two programs. This architecture could be used to create a system with a client application and database server. The database server would process data requests which it has received from the client, and then send the requested records back to the client application. This is a perfectly viable solution for the implementation of this system, however it would be a complex process to write client and server programs and implement their communication using network protocols, a notoriously tricky aspect of programming. As the system must be "always on" and accessible from anywhere this would mean a relatively complex system. This appears to be redundant when powerful server side scripting languages and web servers exist that can be used to implement a system of this type.

2.1.2 Distributed Systems

A distributed system is a computer system composed of large numbers of computers connected by a high speed network [1]. Another way of describing similar systems is to call them "concurrent." A concurrent system must handle separate activities which are in progress at the same time. To be a little more precise, we shall consider two activities to be concurrent if, at a given time, each is at some point between its starting point and its end point [2]. The distributed system which I am going to implement will be a database system with web based interaction. There will also have to be some code written in whichever technology to link the two.

2.2 Technologies

There are many different technologies available to a developer wishing to create a web based system such as the one I have created. The choice of technologies and languages to use can be as important as the implementation, as discovering limitations or flaws in a language during the middle of production can be extremely costly. For this reason I spent a while researching the best options for my implementation

2.2.1 Database Technologies

The choice of database system, although not the most vital of the choices, was still important. One of the key constraints of the project brief was that it should be built using free and open-source software as much as possible. This meant that many of the commercially available DBMS's, such as

Oracle, Access and Delphi, would not be considered. This left me with a choice from the two largest free/open-source programs, MySQL, and Postgres.

2.2.1.1 MySQL

MySQL is available free under the GNU General Public Licence (GPL) and is currently the most widely used open source database server in the world [3]. Its main selling point is the fact that its platform independent, which makes it an extremely useful tool for large systems. However it is very limited in its capacity for complex operations. MySQL does not support sub queries, foreign key queries, views, triggers, unions, or constraints. The lack of these extras makes MySQL significantly faster than PostgreSQL.

2.2.1.2 PostgreSQL

PostgreSQL is the main Linux DBMS and is open source. It can be configured to work on windows OS also, but this requires it to access a UNIX server using TCP/IP [4]. It is a much more powerful DBMS, but sacrificing speed for power, means that it is significantly slower than MySQL. It makes up for this lack of speed by being a much more powerful tool, allowing users to perform all the actions listed above which MySQL cannot.

2.2.1.3 Conclusion

As the system, at least to begin with, is relatively small, it was decided that the speed factor of MySQL was not too much of an advantage as calculations would be kept to a minimum. Although database design had not been begun at this point, it was assumed that the extra functionalities offered by Postgres would be an advantage. The decision was therefore taken to implement the underlying database in Postgres.

2.2.2 Server-side Scripting Language

Server side scripts (SSS) reside on, and are executed on the server. The script can query the database, collect the results and insert them into HTML tags to be sent to the client. There is also the advantage that the server can be located anywhere. SSS provide a wider range of programming options than are available on the client side: for example a server side script can access the server's file structure whereas a client side script cannot [5]. Therefore I decided that a server side scripting language would provide the necessary tools for this project. A large number of possible languages exist, but again the free/open-source consideration meant a limit on which ones were suitable. The languages that were chosen as possibilities are evaluated in the following sections.

2.2.2.1 Perl

Perl is an open source high level programming language that is commonly used for world wide web programming and CGI scripts. By now (2004) it is a relatively old language. The first edition was developed in 1987 and it is now on its 6th version. It is based upon C and therefore should not be difficult to learn. It is a general purpose language and therefore is not web specific. Perl is probably most suited to CGI scripting [6], but was ultimately dismissed as a possibility due to the fact that newer technologies exist.

2.2.2.2 Python

Python is an *interpreted, interactive, object-oriented* programming language. It is often compared to TCL, Perl, Scheme or Java. It combines remarkable power with very clear syntax. It has modules, classes, exceptions, very high level dynamic data types, and dynamic typing [9]. Its possibilities however are not as far reaching as some of the other languages being considered. Java has a whole family of different derivations for web usage, and languages such as PHP were written specifically for web usage. Ultimately then, python is probably not a suitable language for this project.

2.2.2.3 PHP

Hypertext Pre-processor (PHP) is an HTML embedded scripting language that gets much of its syntax from Perl, C and Java [8]. It was written specifically with a view to creating dynamic web pages, and therefore provides excellent speed and functionality. HTTP concepts are built in, which means that detailed knowledge of its headers is not required, as well as having GET and POST variables automatically created for you[8]. PHP can be used with both kinds of DBMS discussed above, and so is therefore an excellent candidate. Its only downfall is that fact that's its early implementations were very insecure. However the latest releases of PHP, v7 onwards, have greatly improved its security, making it a viable choice one more.

2.2.2.4 The Java Family

Java technology was created as a computer programming tool in a small, anonymous, secret effort called "the Green Project" at Sun Microsystems in 1991. It was initially intended to be used for a small, hand held home entertainment device controller, targeted at the digital cable television industry. However it was not accepted as it was too far ahead of its time. Luckily the web was ready for it, and in 1995 it was introduced. Since then it has become perhaps the most used language in the world, being used by more than 4 million software developers in the last 9 years [10]. Since then there have been a multitude of various java based systems to facilitate the programming of distributed systems. These include Jini, Java RMI, JNDI and JavaBeans, among others. There is also the option

of writing ActiveX applets in java, but as these are dependant on the client browser having the ActiveX technologies installed, they were ultimately dismissed as an option.

As java is such a popular language, is platform independent, is very powerful, and I am already familiar with it, it would seem an ideal language to use for implementing the system

2.2.2.5 Conclusion

My research into these technologies threw up two clear candidates, Java and PHP. Ultimately I chose PHP because of the fact that it has commands for cookie setting, DBMS access and other useful properties already available. The final point which swung my decision was my desire to learn as much about web technologies as possible. As I had already studied java for 2 years I was keen to learn something new. For these reasons I chose PHP as my server-side scripting language.

2.2.3 Web Design

There is only one choice of base language for the design of the web front end: HTML. HTML is an extremely diverse language which is capable of creating extremely good looking web pages. When combined with PHP it can produce very dynamic pages, and its lack of competition in the web design world made it an obvious choice.

2.2.4 User Tracking

HTTP is stateless as it processes each message with no knowledge of the previous web page [14]. Due to the stateless nature of HTTP a method of checking whether the user is valid before displaying each page needs to be implemented.

During the implementation phase of the project, it was realised that there would be a need to authenticate and track users through the system. There were 2 main methods of doing this considered; sessions, and cookies.

2.2.4.1 Sessions

In terms of time, a session is the duration of a user visit. In the programming world, a session is an ethereal blob that can hold all sorts of variables and values[15].

Each session object has variables registered with it. Inside the session file, on the server, the registered variables and their values are kept safe and sound. Since these values and variables are not kept in a database, no additional system resources are required[15].

This would seem to be an excellent way to keep track of the users of my system.

2.2.4.2 Cookies

Cookies are a general mechanism which server side connections (such as CGI scripts) can use to both store and retrieve information on the client side of the connection. The addition of a simple, persistent, client-side state significantly extends the capabilities of Web-based client/server applications[16]. This is exactly what we want – the capability to store information about current users.

2.2.4.3 Conclusion

Although sessions are the slicker of the 2 methods of solving this problem, they are also the more complicated. Add this to the fact that the web server on which the system is based seems to have compatibility problems with them, then the decision to implement cookies was taken.

2.3 Past Projects

The reading of previous projects which involved the creation of various database driven distributed systems would be an invaluable aid when creating my own. Many students in the past have had to create similar systems, and their research would be a valuable foil to my own. [23] and [24] were the most useful to me

2.4 Methodologies

A methodology is a set of working methods[11], and there are a few different methodologies when it comes to software design and implementation. Researching these methods was, although not the most enjoyable part of the project, was still interesting.

2.4.1 The Waterfall Model

The waterfall method is perhaps the most simplistic of the various methodologies; it is certainly the most traditional in structure.. It consists of a set of stages which make up the design and implementation process. It is called the waterfall method because its “phases are carried out in sequential process: [12]” the momentum of the project takes you down the steps, and you cannot “go back” and change a previous step after you have moved on. See figure 2.4.1 below.

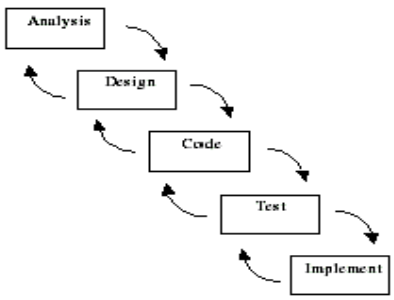


Fig 2.4.1 Waterfall Method of System Development

2.4.2 The Spiral Model

The spiral method offers a much less rigid way of developing a system, as it allows for movement between stages, for example if upon implementation a design flaw is found, it's no problem to go back and change the design. However its main problem is there is no obvious "end," meaning one can get caught up in its cyclical nature, never being satisfied with the system. See figure 2.4.2 below.

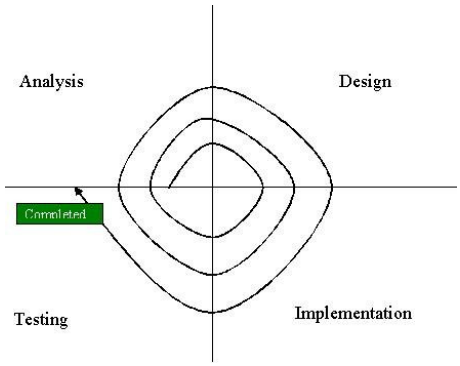


Fig 2.4.2 Spiral Method of System Development

2.4.3 The 'V' Model

The V model is similar in ideals to the waterfall method, however it has been adapted to allow for greater testing of a system. It has similar stages to waterfall, but after every stage there is a parallel testing stage. This means that any bugs or errors are quickly spotted and removed, rather than leaving testing to the end and having any kind of problems occur. See figure 2.4.3 below

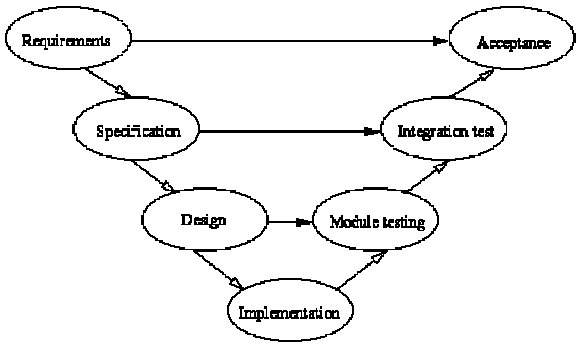


Fig 2.4.3 V Model of System Development

2.4.4 Rapid Application Development (RAD)

RAD was developed by James Martin, in response to what he saw as a problem – that not all requirements are known in advance of beginning a project. It is more focused on the management of a

large group of people, rather than a loan programmer, so was dismissed as a potential development method.

2.4.5 Conclusion

It was decided that non-strict adherence to the spiral model would be the best method to develop this system, as it allows for changes in design during creation. My approach to creating programs and systems has always been to jump in at the deep end and take it from there, so I thought this model would allow me to begin with creation of a test DBMS and set of PHP files. However I decided not to stick to it too strictly, and to work on whichever sections I saw fit at the time. Basically, as a loan developer, I thought it best not to constrain myself too much by committing to a system.

2.5 Human Computer Interaction (HCI)

Web design is a central theme to this project, and as such the Human/Computer Interaction aspect must be considered to be just as important as the rest of the system. If the target users cannot use the system, then it doesn't matter how brilliantly designed or implemented it is. Add to this the fact that as athletes will have spent a greater proportion of their lives doing physical activities, rather than using computers, and therefore may not be as computer literate as they could be. For this reason the system should be designed to be "idiot proof," meaning that it is clear what to do at all times.

2.6 Functional Requirements

The following were identified as the key functional requirements of the system. The system must:

1. Store data on athletes and their coaches, as well as who is coached by whom.
2. Store data on athletes training and times and injuries.
3. Allow athletes to store and view their times.
4. Allow athletes and coaches to send messages to each other.

Chapter 3: Design

The architecture used as a basis for building the system is detailed in this chapter.

3.1 Overview

By this point we have a fairly good idea of how our system is going to be implemented. We are going to have a Postgres DBMS, and a web based user interface which utilises PHP to connect and perform actions between the two. As design progressed, additional functionalities were realised, and added. The main areas of design were

- DBMS
- PHP scripts
- Web Front End

3.2 Structure of the System

The deployment of the system is shown below in figure 3.2. The system is split into client and server, with the 1st tier on the client, and the 2nd and 3rd tiers on the server.

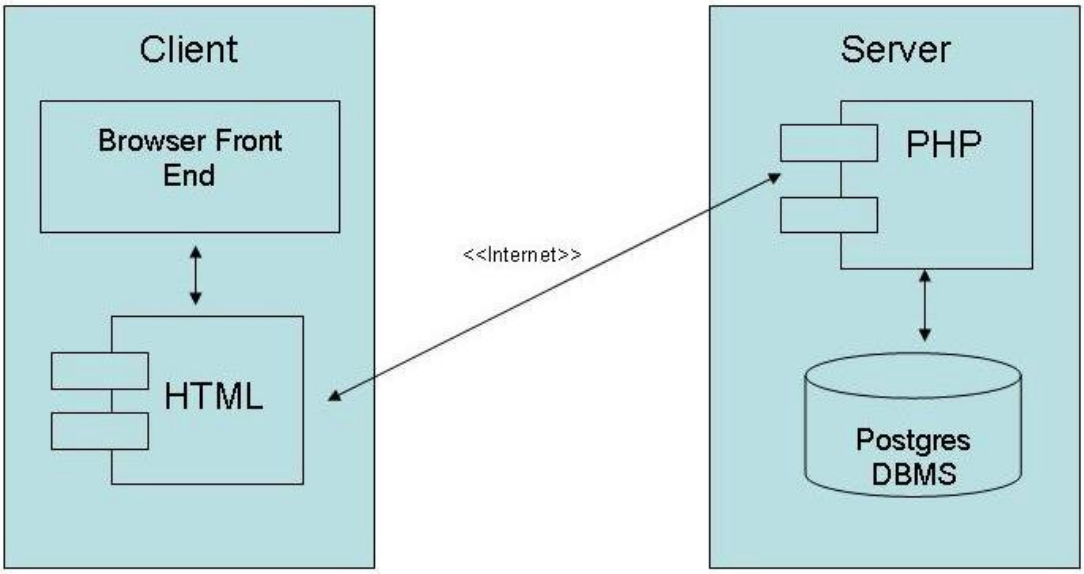


Figure 3.2 Proposed Deployment and components of the system

The design of the 3 main sections of the system will be as follows:

3.2.1 Database

The Postgres DBMS was the first bit of the system to be designed. As this is where all relevant information is going to be held, it was vital to know exactly what pieces of information it was expected to store. Primary goals of the DBMS are:

- Store basic information on athletes, such as name, events in which they compete etc.
- Store athlete login information, so as to have something to authenticate login attempts against.
- Store results in events for which athlete is training.
- Store data on past or current injuries.
- Store basic information on coaches, such as name, who they coach etc.
- Store coach login information, so as to have something to authenticate login attempts against.
- Store messages between coaches and athletes.

Clearly, separate tables would be needed in the DBMS for each of these, and probably more.

Elmasri and Navathe describe the importance of normalized databases so as to minimise redundancy and insert, update, and delete anomalies [13]. The importance of normal form is something which has been reiterated many a time throughout database modules, and was something I fully intended to honour in my design.

I was confident that, despite the fact that there is quite a lot of different information to store, the system would actually be relatively simple. The design of the tables is shown below, primary keys are underlined, with a description of the information they store underneath

athlete(ath_id:int, surname:varchar(30), forenames:varchar(30))
Stores ID, name and surname of athletes

coach(coach_id:int, surname:varchar(30), forenames:varchar(30))
Stores ID, name and surname of coaches

coached_by(ath_id:int, coach_id:int)
Shows which athletes are coached by which coaches

event(event_id:int, name:varchar(30))
Stores unique Ids and names of events.

competes_in(ath_id:int, event_id:int)
Shows which events athletes compete in

results(ath_id:int, event_id:int, score:varchar(30))

Stores results for athletes for their events

injury(ath_id:int, start_date:date, end_date:date, injury:text)
Stores athlete injuries and the dates they were injured for. If end_date is null, injury is current

ath_access(ath_id:int, password:varchar(30))
Stores passwords for each athlete

coach_access(coach_id:int, password:varchar(30))
Stores passwords for each coach

athlete_message(id:int(auto increment), ath_id:int, coach_id:int, message:text)
Stores athletes messages, as well as who they were sent by

coach_message(id:int(auto increment), coach_id:int, ath_id:int, message:text)
Stores coaches messages, as well as who they were sent by

Some tables don't have primary keys, because they don't need them. For example the "competes_in" relation is a many to many relation, therefore needs no primary keys. As it was such a simple system, it was decided that foreign keys were not needed. At any one time joins would only be

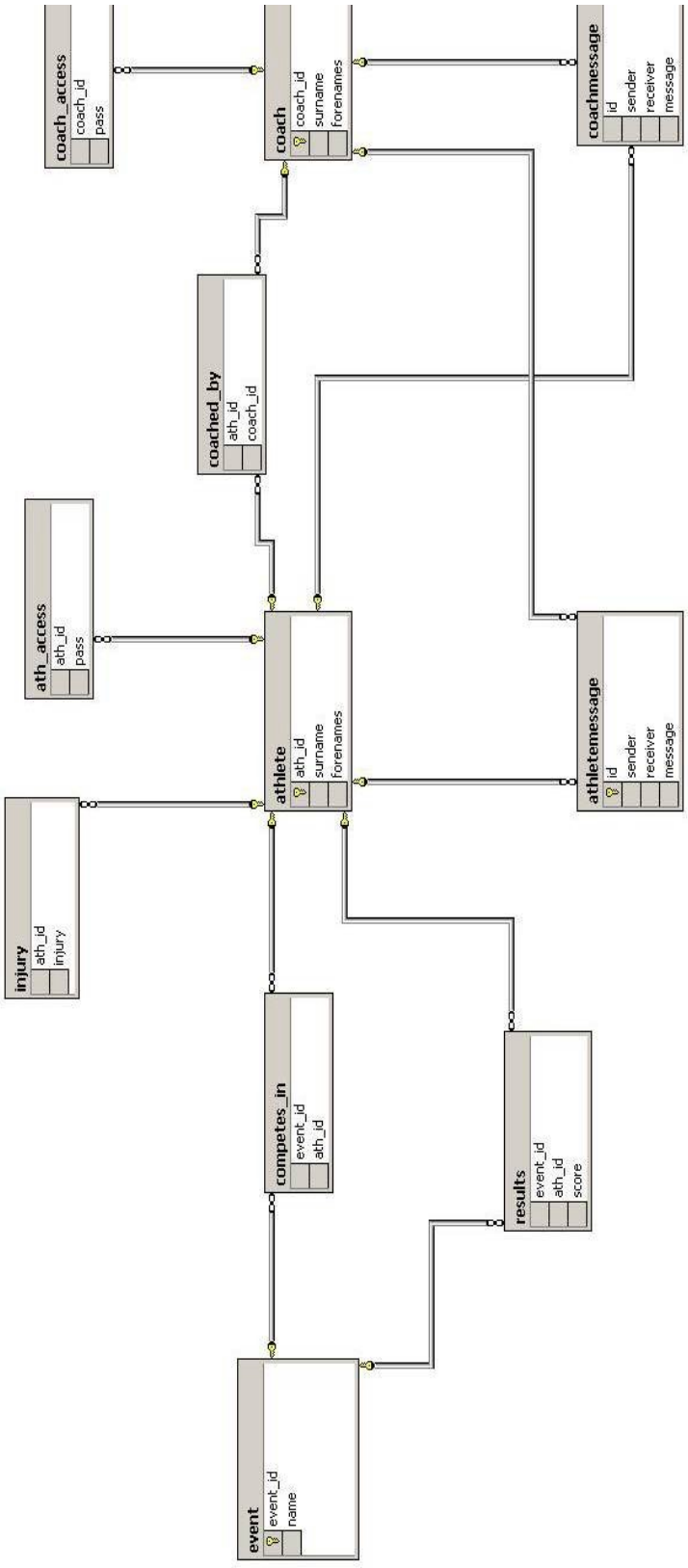


Fig 3.2.1: ER Diagram for the Database Schema

required between, at most, 3 separate tables. By keeping the names of the ath_id and coach_id fields constant throughout the tables, it was realised that "Natural Joins" could be used, further simplifying the DBMS.

3.2.2 User Interface

The design of the user interface was one of the most important aspects of the system. Professional chef's use the phrase "the first bite is with the eye," meaning that regardless of how good what you have created is, if it doesn't look good then you are already behind. The same is going to be true for my system. If users found it ugly there were less likely to use it.

Fortunately HTML provides very simple and clear methods of creating buttons, forms, and other useful tools. These are what would be used for all information gathering aspects of the system, such as login pages and time entering. There would clearly have to be a mixture of static HTML and dynamic PHP pages required for the various functions the system would have to provide. These pages would be required as follows:

- index.html – base page required as a welcome page, with links to login scripts
- Login – html page to input information, PHP script to validate sed information against the database. Set cookies. Separate pages and scripts for coaches and athletes.
- View general info – PHP page to retrieve general information such as name, and who coaches you/who you coach, depending on user.
- View injuries – PHP script for athletes and coaches to retrieve injury data on themselves, or the athletes they coach.
- Add injuries – HTML form and PHP script for athletes to add new injuries to the database.
- View times – PHP script for athletes to retrieve their own info, HTML form for coaches to enter name of athlete they wish to view. PHP validation that coach does coach athlete.
- Add times – HTML form, PHP validation/insertion
- Send message – HTML form, PHP validation/insertion plus delete capability
- Read message – PHP retrieval and display
- Logout – PHP script to delete cookie, reload index page.

It is expected that additional requirements and functionalities will be thought up along the way, as well as different ways of solving the same problems. This is one of the reasons that the spiral model was chosen as the best method of development.

I have been designing and implementing various web systems since I was 15, and so was extremely confident with this area of the project. Because of this, the looks of the site were not subject to the same design principles as the rest of the system, as I intended to create these on the fly, as I always have in the past.

3.2.3 PHP Scripts

The PHP scripts will be written on the fly depending on their functionality required. SQL statements will be included to allow for database insertion and extraction. For details on PHP scripts, please see the implementation chapter.

3.2.4 User Tracking/Cookies

The decision to use a web based system means a web browser will be used to provide an interface to the users. The HTTP protocol will be used to transmit messages passed to the server. It was not until during the implementation of the web system that the ramifications of the fact that "HTTP is a stateless protocol[14]" were fully realised: variables will need to be passed between each page in order to create the impression of a connection-oriented protocol, such as when a coach wishes to view one of his athletes' times. The method of doing this that was decided upon was to implement client side cookies.

Cookies in PHP are extremely simple to use. PHP has built in functions which create and read information from them, meaning that all I had to do was invoke them with the correct parameters. For full details on this see the implementation chapter:

Chapter 4 Implementation

As stated before, the adopted methodology, the spiral method, allows for fluid movement between stages. especially in the design, implementation, and testing stages. The implementation started shortly after the design was begun, and progressed with each stage of the design. The testing (see chapter 5) took place during and after the implementation. The design was flexible and allowed for change in the event of technological constraints discovered through implementation (such as occurred with the intended session implementation).

4.1 Database

The Postgres DBMS was created according to the schema set out in 3.2.1. It was also populated with a small amount of test data. It was managed through the PSQL program on the School of Computings Linux machines, and was located on their CSDBDEV server. PSQL is a simple text based DBMS program.

SQL statements were required to be embedded in the PHP. These were reasonable basic, with only insert, update, delete and joins being used.

4.2 HTML/User Interface

I originally intended to use XML and Cascading Style Sheets, but decided that such a dynamic approach was not required for this system. Such an approach would normally be taken by someone developing a large website, which he would like to change the look of periodically. Since this is a static system I decided to stick with the standard HTML approach. The other major decision was the one to not use frames, but to code the various menu's and side, top and bottom bars into every web page. This was merely my own preference, as I feel that frames make a page look very separated, as well as being displayed differently in different browsers.

To still give the impression of a structured page, it was decided to use the <table> and tags to keep menus and pictures, as well as text areas correctly aligned. This produces an effect similar to frames, but without the separated look.

4.3 PHP

The development of the PHP, as with the rest of the code development, followed the same methodology as the rest of the project, allowing for changes is design due to problems or idea's discovered during implementation. It was not begun, however, until I was happy with the looks of the HTML part of the site. Once I was happy with how it would look, I moved on to embedding the PHP.

When I came to begin my PHP implementation, it was the first time I had used the language. Although I was confident that my experiences with other Object Oriented Languages, it was clear that I would need some practice to get used to the quirks and individual way of handling certain operations which every language has. As PHP is a web scripting language, the vast majority of resources on it

are located online. In teaching myself the required bits of this language, [15]. [17] and [18] were invaluable. Although these sites were used to teach myself the language, I must point out that none of the code is taken from them, it is all original work.

Pretty much every PHP file created was to read data from or write it to the DBMS. PHP already has built in functions which give it access to Postgres databases. This function is invoked as follows:

```
$dbcon=pg_connect("host=csdbdev.leeds.ac.uk user=scs1jdm
password=xxxx dbname=scs1jdm");
```

Once invoked the $dbcon variable becomes a global variable, meaning that once the DB connection is established, it need not be referenced, it is automatically used.

4.3.1 Retrieving Data

Retrieving data from a DB is slightly more difficult than first thought. Rows are returned via the pg_exec command, which executes SQL against the connection created above. To retrieve the data, first we must assign it to a variable:

```
$RowArray = pg_exec(“SQL statement”);
```

The data now stored in $RowArray is still the raw data, still with the Postgres header around it, no good for display or comparison. Say we are trying to retrieve a password from the DB for comparison, we would need to properly extract the string from the array:

```
$rawPass = pg_fetch_array($RowArray);
$password = $rawPass["pass"];
```

$password is now correctly formatted for comparison, or display.

4.3.2 Inserting Data

Inserting data is easier. PHP allows variable insertion into strings without the need to break the strings up, meaning insertion of data is easier to implement. All that was needed was to call pg_exec on a suitable SQL statement:

4.3.3 Page By Page

This is just a simple run-down of what each page, along with its embedded PHP scripting, is supposed to do.

index.html: base page, welcomes user to the system, and provides 2 links for logging in; 1 for athletes and 1 for coaches.

athleteLogin.html and *coachLogin.html*: provides 3 text input boxes with suitable prompts asking for Surname, Forenames and passwords. Sends these to *athleteLogin.php* and *coachLogin.php* depending on which is required.

athleteLogin.php and *coachLogin.php*: takes usernames and passwords. Retrieves actual password for given user from the database and compares it with the one entered. They have 3 different outcomes:

1. User is not in database, error message given and sent back to login page.
2. Password is incorrect, error message given and sent back to login page.
3. Username and Password are correct, success message given, *createCookie.php* or *createCoachCookie.php* called with the relevant ID passed to it.

createCookie.php and *createCoachCookie.php*: these simply create a cookie storing the coach or athletes unique ID allowing the system to identify them. They then point the browser to either *welcomeAthlete.php* or *welcomeCoach.php*.

welcomeAthlete.php and *welcomeCoach.php*: these are the welcome pages for users once they are logged on. It uses the ID cookie to access forenames and messages, so it can display “welcome X, you have X messages to read”

getInfo.php and *getCoachInfo.php*: shows general info about the user of the system. Athletes see their name, events which they compete in, and who coaches them. Coaches see their name, and who they coach.

getTimes.php: retrieves all results for all events for the current athlete, and displays them in a table.

enterTimes.php: retrieves the events the current athlete is registered for and puts them in a drop down box. Text input for score, and selected event and score are sent to *enterTimesToDB.php*.

enterTimesToDB.php: takes the event, time, and athlete id from the *enterTimes.php* form, and inserts the info into the results table. Then points the browser to *getTimes.php* to show the new time.

getCoachTimes.php: version of getTimes.php for the coaches. Has a form for athlete surname and forenames, passes the entered info to *getCoachTimesFromDB.php*

getCoachTimesFromDB.php: takes the athlete names and performs 1 of 3 actions:

1. If athlete is not in the database, a suitable error message is sent
2. If coach does not coach that athlete, a suitable error message is sent
3. If coach does coach athlete, and athlete is in the system, then it displays times for that athlete.

readAthleteMessages.php and *readCoachMessage.php*: retrieves all messages for the current user from the relative table – coachmessage or athletemessage, and displays them in a table.

sendMessageToAthlete.html and *sendMessageToCoach.html*: provides a simple form for intended recipient and message. Passes recipient and message to *sendMessageToAthlete.php* and *sendMessageToCoach.php*

sendMessageToAthlete.php and *sendMessageToCoach.php:* takes the recipient and message as argument. If recipient is in the DB, message is written to the relevant table, else a suitable error message is sent.

deleteAthleteMessage.php and *deleteCoachMessage.php*: called from a link in *readAthleteMessages.php* and *readCoachMessage.php*, takes a message ID as an argument (ID provided by the delete link with no action needed from user) and removes that message from the database.

logout.php: Deletes cookie and reloads login page, ready for a new user.

4.4 Security

Although there is no sensitive information passed, such as bank details, or criminal or medical records, security still needs to be addressed as this is a web based system that will be subject to scrutiny. PHP is a powerful language and the interpreter, whether included in a web server as a module or executed as a separate CGI binary, is able to access files, execute commands and open network connections on the server. These properties make anything run on a web server insecure by default. PHP is designed specifically to be a more secure language for writing CGI programs than Perl or C, and with correct selection of compile-time and runtime configuration options, and proper coding practices, it can give you exactly the combination of freedom and security you need[19].

Some basic security was implemented, such as password masking on the form, and using POST instead of GET to send/retrieve them, making them hidden. No athlete or coach can view information he/she is not supposed to have, without resorting to hack techniques such as SQL injection.

As is usual with development of computer systems, security was traded off for usability and additional functionalities. Security was not seen as an important aspect when compared to the additional functionality requirements. For more info on possible security features, see Evaluation.

4.5 Implementation Problems

As has been mentioned before, the spiral methodology was adopted to allow for constant revision of the design during implementation. There were a few things that were intended which turned out to either not be possible, or to take up too much time. Also some of the features which have been implemented caused many problems.

4.5.1 User Authentication

The intended method of authenticating users was to use the $PHP_AUTH_USER global variable to send the 'HTTP/1.0 401 Unauthorized' header if it was not set, as detailed in [15]. By using the standard HTTP headers and dialogue boxes it was hoped to gain some better security, however the web server didn't agree with me sending header in this fashion, and refused to set the $PHP_AUTH_USER variable. This lead to the actual implementation, which was to use HTML forms.

4.5.2 User Tracking

The intended method of user tracking was to use session cookies to provide a slicker method of tracking than cookies provided. However the web server seemed to have them disabled for security reasons. This meant that I would have to use cookies to track users of the system.

One major problem that was encountered when implementing the cookies was the fact that when they were tested in Internet Explorer 6, they didn't work. Implementation of the cookies had been taking place on the Linux machines in the School of Computings laboratories, with Konqueror as the browser. Once it came to browser compatibility testing it was discovered that the cookies did not work in IE6. As IE is the worlds most popular browser this was disastrous to the stage of the project. It took a further week during final testing to discover the error and re-implement.

4.5.3 Conclusion

I'm sure that with more time spent on the PHP aspect of implementation, and with my own web server, that I could implement the 2 problems I encountered in the fashion in which I wanted. PHP is still a relatively new scripting language meaning that there is not the wealth of support available for it, both from university support and teaching staff, and from the online community.

Chapter 5 Testing

As mentioned many times before, the methodology adopted for this project is the spiral model, which doesn't have clear cut stages. This meant that testing occurred pretty much as

development proceeded. This led to a very stop start approach to development and testing, as I only moved onto the next bit of implementation once I was satisfied that the previous bit was working correctly.

5.1 System Testing

The following table of testing was carried out after all development was complete.

Test	Method	Pass Criteria	Result
Does index page load correctly?	Point browser to root directory of system	Page loads correctly	Pass
Does athlete login page load correctly?	Click on athlete login link on index page	Page loads correctly	Pass
Does coach login page load correctly?	Click on coach login link on index page	Page loads correctly	Pass
Are valid athletes allowed to log in?	Attempt to log in using valid information: “Bell”, “Catherine”, “abc123”	Athlete is logged in successfully with “You’re Authorised” displayed	Pass
Are valid coaches allowed to log in?	Attempt to log in using valid information: “France”, “James”, “sanan”	Coach is logged in successfully with “You’re Authorised” displayed	Pass
Are invalid athletes refused entry to system?	Attempt to log in using invalid information: “Bull”, “Catherene”, “123abc”	Athlete is refused entry to system with “Invalid info, please try again” displayed	Pass
Are invalid coaches refused entry to system?	Attempt to log in using invalid information “England”, “Jim”, “ansan”	Coach is refused entry to system with “Invalid info, please try again” displayed	Pass

Test	Method	Pass Criteria	Result

Are cookies set?	Login using valid information. Check for cookie on computer.	Cookie is present for both athletes and coaches	Pass
Do set cookies contain correct information?	Log in and check that correct details are displayed	Correct name and data displayed in “general info”	Pass
Are cookies removed successfully on log off?	Log off then re-load one of the system pages	User directed to log in page	Pass
Correct info displayed on “getInfo.php” and “getCoachInfo.php”	View pages for all athletes and coaches respectively	Correct info displayed	Pass
Correct info displayed in “View Times”	Display times and distances for all athletes	Only the logged in athletes results shown	Pass
Only events that the athlete participated in are displayed on in drop down box “enterTimes.php”	View “enterTimes.php” when logged in as all users and check the drop down box	Only events for which athletes are registered are displayed	Pass
New times or distances correctly entered by “enterTimesToDB.php”	Enter a new time for each athlete then view their times	Newly entered info correctly displayed	Pass
“sendMessageToCoach.php” correctly identifies invalid coach names	Try to send a message to non existent coach “Dave”, “Gorman”	Message “Coach not found, please try again” displayed	Pass
“sendMessageToCoach.php” will not let athletes send messages to coaches that do not coach them	Try to send message to coach who does not coach athlete we are logged in as	Message “This person is not your coach” displayed.	Pass
“sendMessageToCoach.php” correctly sends messages to coaches	Try to send message to real coach: “Davies”, “Simon”, “testing”	Message correctly written to coachMessage database	Pass

Test	**Method**	**Pass Criteria**	**Result**
“sendMessageToAthlete.php” correctly identifies	Try to send a message to non existent athlete “Dave”,	Message “Athlete not found, please try again”	Pass

invalid athlete names	"Gorman"	displayed	
"sendMessageToAthlete.php" will not let athletes send messages to coaches that do not coach them	Try to send message to athlete whom coach we are logged in as does not coach	Message "You do not coach this athlete" displayed.	Pass
"sendMessageToAthlete.php" correctly sends messages to coaches	Try to send message to valid athlete: "Bell", "Catherine", "testing"	Message correctly written to athleteMessage database	Pass
"readAthleteMessages.php" correctly displays messages for current athlete.	Enter a message for the athlete into the athleteMessage database and check it is displayed	Message correctly displayed	Pass
"readCoachMessages.php" correctly displays messages for current coach.	Enter a message for the coach into the coachMessage database and check it is displayed	Message correctly displayed	Pass
"deleteAthleteMessage.php" correctly deletes the selected message from the database	Click on "delete" next to message on system	Message correctly deleted	Pass
"deleteCoachMessage.php" correctly deletes the selected message from the database	Click on "delete" next to message on system	Message correctly deleted	Pass

5.2 HCI Testing

Usability is crucial to the success of any software. Usability addresses the relationship between tools and their users. In order for a tool to be effective, it must allow intended users to accomplish their tasks in the best way possible. The same principle applies to computers, websites, and other software. In order for these systems to work, their users must be able to employ them effectively [20].

The ability for other people to use the system was extremely important. For this reason I had 10 people complete a questionnaire on the usability of the system. These people included fellow

computing students, students of other departments, and older people with little or no experience with computers. The questionnaire can be seen in appendix [INSERT HERE].

As can be seen from the questionnaires, results were extremely favourable. Everyone said that the system was aesthetically pleasing, and that every required action was clearly labelled. I have included a copy of the questionnaire below, with average values taken from the results. All questioned users used the system in Internet Explorer 6.

How would you rate the looks of the system?	4.3
How easy was it to log in as an athlete from the index page?	3.9
How easy was it to log in as a coach from the index page?	3.9
How easy was it to add new times to the system?	4.3
How easy was it to view your times?	4.6
How easy was it to send messages?	4.2
How easy was it to read messages?	4.6
How easy was it to log off?	4.6

5.3 Browser Compatability Testing

Despite the fact that HTML and PHP are platform independent, the clients browser can be an extremely important factor with usability, as they can display content differently, and even treat header information differently.

Internet Explorer versions 5 and 6 count for more than 80% of total internet usage. Mozilla has 9.6%, and Netscape Navigator versions 3, 4, and 7 together add up to 2.2% of usage. Opera has 2.1%. AOL also has a small percentage, but the figures are not recorded for 2004 [21]. With these statistics in mind, it was seen as important to test the web system in certain browsers. The following table gives the results of this testing:

Browser	**Result**
Internet Explorer 6.0	Works Correctly
Internet Explorer 5.0	Works Correctly
Konqueror	Works Correctly, some layout errors
Mozilla	Works Correctly, some layout errors

Chapter 6: Evaluation

Evaluating ones own work is one of the most important skills the developer of a computer system can have. There is a well known phrase that goes along the lines of “if we do not learn from

our mistakes, we are doomed to repeat them." This is exactly the reason why (unbiased) evaluation is such a vital skill for a software programmer.

I also feel that it is important for a system like this to have some kind of heuristic evaluation performed on it, as it is going to be used by a great many people. In general, heuristic evaluation is difficult for a single individual to do because one person will never be able to find all the usability problems in an interface[22], particularly if he is the person who created that interface.

6.1 Evaluation Criteria

At the outset of the project a set of minimum requirements, and minimal functional requirements were set out. This project will be deemed a success if these criteria are met.

6.1.1 Minimum Requirements

For the project to be successful it must meet its minimum requirements. They are repeated below, follow by an analysis of whether they were met.
The Minimum Requirements were:

6. Understand athlete's training systems & gather basic user requirements for the system
7. A Database capable of storing all required information
8. A Web page with a form to enter new training regimes and times
9. Some basic CGI to take the form's information and insert it into database tables
10. Some basic CGI capable of querying the database, and displaying returned rows in a suitable format.

Analysis:

1. was met, as user requirements were gathered through reading [23], and an understanding of how athletes train was garnered.
2. was met, as the database is capable of storing all required information.
3. was met, as there is an extensive web based system for inserting and viewing information.
4. was met, as the PHP scripts written insert data without problems.
5. was met as the PHP scripts written can return required information, and display it to the screen.

6.1.2 Functional Requirements

The other criteria to evaluate by were the functional requirements. They were:

5. Store data on athletes and their coaches, as well as who is coached by whom.

6. Store data on athletes training and times and injuries.
7. Allow athletes to store and view their times.
8. Allow athletes and coaches to send messages to each other.

These 4 main functional requirements have all been met, as can be seen from the testing section.

6.2 Methodology

The methodology used for this system was the spiral method of system development. Adopting this methodology was, in the authors mind, the best decision taken during this project, as it allowed complete freedom when designing, implementing and testing the various different parts of this system. The ability to move between stages at will, meant I could work on different sections at a time.

6.3 Project Management

The schedule drawn up at the beginning of this project should have acted as a valuable guide in allowing the author to remain in control of the project. The progression of the project however did not go as smoothly as was predicted in the schedule. Because of the methodology chosen, the actual schedule taken is not really possible to define, as the design, implementation and testing phases were all interwoven. Added to this was the fact that that the system turned out to be complex in nature than was first thought. This had the knock on effect of delaying the writing of the report. On reflection, a re-design of the schedule once the spiral method had been adopted, may have been an advisable move. The usability tests also made some additional requirements and interface issues apparent that required additional implementation. A buffer of a few weeks was deliberately left at the end of the project to allow for any stages which overran, which allowed the delays to be managed and the project to be delivered on time. Just.

Flexibility was vital to the project in the sense that it allowed the implementation and testing stages to be carried out in parallel. Each part could be tested separately, and then in integration with the rest of the system. The fact that the method of performing each stage required is left to the individual was ideal. It meant that the tools to be used at each stage of the project could be used in response to the requirements gathered and implementation and testing carried out.

6.4 Heuristic Evaluation

Heuristic evaluation is performed by having each individual evaluator inspect the interface alone. Only after all evaluations have been completed are the evaluators allowed to communicate and

have their findings aggregated[22]. A slightly less formal method of evaluation was taken for this than the HCI testing, as it was felt that as HCI testing was successful, that there could be little wrong with the heuristics. I had several friends from the School of Computing, as well as 2 from outside the school use the system, and aside from the security concerns mentioned in chapter 7, nothing was found to be wrong.

6.5 Evaluation Summary

The athlete training system was preceded and complemented by background research into the various areas of technology and computing that are involved. It took on a flexible methodological approach, gaining from aspects of the Spiral and Iterative Methods. The design and implementation consisted of applying HCI and usability techniques to the user interface, functionality to the PHP scripting, and SQL knowledge to the database system. The testing and took place alongside the implementation, as well as after it was finished, and the input and output were both tested thoroughly. The evaluation of the project used questionnaires and requirement criteria, which were not only used to evaluate what is there, but also used to formulate the future enhancements in section 7.

Considering that the minimum and functional requirements were met, I would say that the project was a success. Obviously there were failures, as well as successes along the way, but overall I feel the project did what it was supposed to do: give me an idea of how such a project would be managed in the real world of software development.

6.6 Future Enhancements

There are many future enhancements that I would like to make to the system, encompassing both functionality and security.

6.6.1 Functionality

Unfortunately, as with any system, there were additional functionalities which didn't make it into the final system, either because of time or lack of programming knowledge. The additional functionalities I would like to have implemented are:

- The ability for athletes to calculate averages of their scores. This would be useful as athletes could see their "rolling averages," as well as keeping track of overall improvement.
- The ability for athletes to generate graphs of their scores. This would be useful as athletes could see trends in their scores, allowing them to tune their training to improve certain areas.
- The ability for athletes to generate reports on their injuries. This would be useful in spotting injury trends.

6.6.2 Security

The additional security requirements I would like to have implemented are:

- The login system is not as secure as I would have liked, I would have liked to have implemented password hashing in order to prevent eavesdroppers.
- The cookies are not secure; I would have liked to have found some way of encoding them, as at the moment changing your ID can give you access to other people's areas.
- SQL injection is a problem as no additional input validation was performed. PHP4 does provide some, but not enough. This vulnerability can expose the entire database to anyone who knows how to do it.

Bibliography

[1] Tanenbaum, A. S. *"Distributed Systems: Principles and Paradigms"* 2002.

[2] Bacon, Jean *"Concurrent Systems,"* Addison-Wesley 1997.

[3] MySQL AB, (2003), MySQL, URL: http://www.mysql.com/ 30th Nov 2004

[4] The PostgreSQL Global Development Group, (2004), PostgreSQL,
URL: http://www.postgresql.org/ [30th Nov 2004]

[5] Deitel HM, Deitel PJ, Nieto TR, "*Internet and World Wide Web How To Program*" Prentice Hall 2002

[6] "Perl.com" Homepage URL: http://www.perl.com/ 1st Dec 2004

[7] "MediaCollege.com" "*Perl vs. PHP*" Accessed 1st Dec 2004
URL:http://www.mediacollege.com/internet/perl/perl-vs-php.html

[8] The PHP Group URL:http://www.php.net/ 30th Nov 2003

[9] "Python.org" URL: http://www.python.org 5th Dec 2004

[10] "Java.com" URL: http://www.java.com/en/about/java_technology.jsp 5th Dec 2004

[11] "dictionary.com" URL: http://www.dictionary.com/ 4th Feb 2005

[12] Avison, D. Fitzgerald, G. "*Information Systems Development: Methodologies, Techniques and Tools*" McGraw-Hill 2002

[13] Elmasri, R. & Navathe, S. B. "*Fundamentals of Database Systems.*" Addison-Wesley 2003

[14] Webopedia , "*What is a Stateless Model*?"
URL: http://www.webopedia.com/TERM/s/stateless.html 20th Apr 2005

[15] Julie Meloni for Webmonkey "Authenticate and Track Users with PHP"
URL: http://webmonkey.wired.com/webmonkey/00/05/index2a.html 1st Apr 2005

[16] Netscape.com "*Persistent Client-Side HTTP cookies Preliminary Specification*"
URL: http://wp.netscape.com/newsref/std/cookie_spec.html 3rd April 2005

[17] w3 Schools "Tutorial on PHP" URL: http://www.w3schools.com/php/default.asp November 2004 - April 2005

[18] DevShed PHP pages URL: http://www.devshed.com/c/b/PHP November 2004 - April 2005

[19] The PHP Group "*Security*" URL: http://uk2.php.net/manual/en/security.php 20th April 2005

[20] Usability.com URL: http://www.usabilityfirst.com/intro/index.txl 21st April 2005

[21] W3 Consortium.. *"Browser Statistics"*. URL:
http://www.w3schools.com/browsers/browsers_stats.asp 21st April 2005

[22] UseIt.com "*How to Perform a Heuristic Evaluation*" URL:
http://www.useit.com/papers/heuristic/heuristic_evaluation.html 24th April 2005

[23] Allen, K "*Database Driven Online System and Analysis for RETAS-Leeds*" University of Leeds 2004.

[24] Hall, G "*An SLTC Meetings Management system*" University of Leeds 2004.

Sources read, but not directly referenced:

Burke, Edmund J. "*Toward an understanding of human performance: readings in exercise physiology for the coach and athlete*" Mouvement Publications 1978

Perrin, David H. "*The injured athlete*" Philadelphia 1999

Davies, C. "*Addition of Dynamic Business Logic to School of Computing Website*" University of Leeds 2003.

Images

Fig 2.4.1 From http://herkules.oulu.fi/isbn9514266005/html/graphic55.png

Fig 2.4.2 From http://www.programmersheaven.com/articles/images/guruprasad/spiral.jpg

Fig 2.4.3 From http://disco.cs.tut.fi/tutorial/v_model_t.gif

All other images are original work by James Molyneux

Appendix A: Personal Reflection

Upon entering the 3rd year of my computing degree I was full of excitement and confidence. I was looking forward to the challenges facing me, not only academically, but also professionally. I saw

my final year project as an opportunity to expand my knowledge of the computing world, and to push myself and really "see what I was made of."

Upon choosing projects, I was extremely lucky. At first I was given none of my choices, only to find out I could pick from any of them. I chose the athlete training system for 3 main reasons:

- I had recently taken, and enjoyed the distributed system modules offered by the School of Computing.
- I had a lot of previous experience in web design, and thought I could create a good looking system.
- I was excited at the prospect of learning new technologies.

Having embarked upon the project, it quickly dawned on me just how much was going to be required. However as I had weighted my year 70:50, meaning I had less modules the second semester, I felt that leaving the majority of the work until the second semester would be a good move, and that merely doing the background reading and research in the first would be enough. How wrong I was.

I quickly discovered the massive scope of my project when I began it in earnest. Not only did I have to completely design and create the system from scratch, but I had to test, evaluate and write it up also. I think that I severely underestimated the amount of time and work that would be required to complete this write-up, and as a result I feel the overall quality of my system has suffered.

However I do feel that it is still a good system. It fulfils my minimum requirements, and I have even implemented an additional messaging system which I feel works well. Overall though, I do see the system as incomplete. There were so many ideas that I had for it which I just never got round to implementing, or just couldn't. if I could do it again I would adopt a different methodology, and spend time working out exactly what I wanted the system to be able to do, enabling me to implement it, and for my system to be all I wanted it to be.

As far as the project goes, I really did enjoy the creation of my system. Even though it was extremely stressful, particularly towards the end, I still feel it was an invaluable experience. Learning a new scripting language in PHP was one of the reasons I took on the project in the first place, and I feel I am now a competent user of PHP. These techniques I have learned, together with the time management lessons will surely be an asset to me as I go about my professional life.

Overall, upon leaving the University of Leeds, my overall feeling is one of regret. If I could change one thing about my academic career, I would change my attitude to the work. Before coming to university I never had any problems with any work that came my way, pretty much leaving it until the last minute and still getting good results. However the massive culture shock of moving away from home and being given complete freedom was my downfall. Coming to this university, and being taught by the huge amount of knowledgeable people was such a massive opportunity, and I just wish I could have made the best of it.

However there were plus points. There is no doubt that I have learned a hell of a lot in my time here, not only technologically, but also about life, love and the pursuit of happiness. I have had an amazing time, and I have made friends here that I am sure will be with me until the day I die. I suppose that that there are plenty of people who would love to have that.

As it is, I will be leaving this institution with what will probably amount to a 2ii, which although is certainly worth having, will probably not allow me to reach the heights which I always assumed I would. However until they invent time machines, I suppose I'm stuck with my past actions, and the only thing to do is to make the best of it.

Thank-you Leeds for what was undoubtedly the best 4 years of my life. I just hope I can do you proud in the future.

James Molyneux

Appendix B Questionnaire on Athlete Training System

Name:

Delete as appropriate: Computing Student / Other Student / Other

Please rate the following questions 1 - 5, 1 being the lowest and 5 being the highest.

How would you rate the looks of the system?	1 2 3 4 5
How easy was it to log in as an athlete from the index page?	1 2 3 4 5
How easy was it to log in as a coach from the index page?	1 2 3 4 5
How easy was it to add new times to the system?	1 2 3 4 5
How easy was it to view your times?	1 2 3 4 5
How easy was it to send messages?	1 2 3 4 5
How easy was it to read messages?	1 2 3 4 5
How easy was it to log off?	1 2 3 4 5

Comments?

Appendix D: User Manual

Setup Requirements

The system is currently running on the SoC web development server. The index page can be found at http://wwwdev.comp.leeds.ac.uk/scs1jdm/

You can log in as an athlete using the following information
Surname: Molyneux or Bell
Forename: James or Catherine
Password: abc123

You can log in as a coach using the following information
Surname: Davies
Forename: Simon
Password: teacher

All HTML files, PHP files, and SQL command files required to set up the system are included on floppy disc.

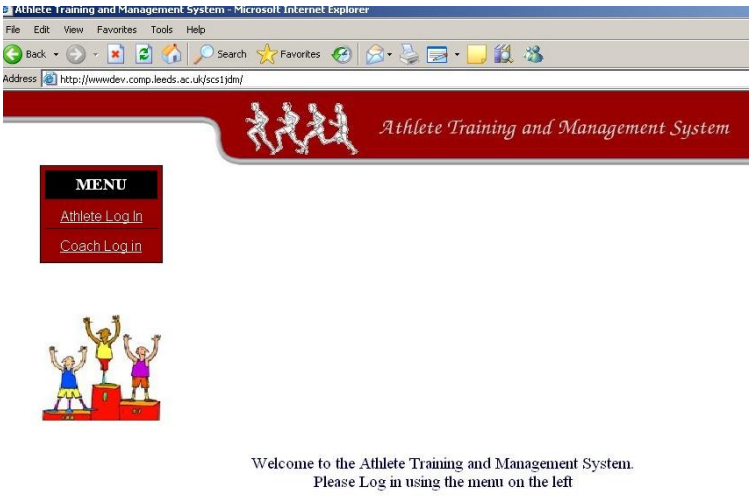
To use the files as they are on your own system, you would need a web server running PHP v 4.0 or later, as well as a Postgres DBMS system running.

The system will run correctly in Internet Explorer 5.0 or later, Mozilla or Konqueror. Usability of all other browsers ahs not been tested.

Athletes

Logging In

Athletes enter the system at the index page:


Athlete Training and Management System - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://wwwdev.comp.leeds.ac.uk/scs1jdm/
Athlete Training and Management System
MENU
Athlete Log In
Coach Log in
Welcome to the Athlete Training and Management System.
Please Log in using the menu on the left

And navigate to their login page by clicking the appropriate link in the menu on the left

Enter the relevant information in the boxes provided. Should you enter your username incorrectly you will see the message

"Athlete Not Found, Please Try Again"

Should you enter an incorrect password, you will see the message

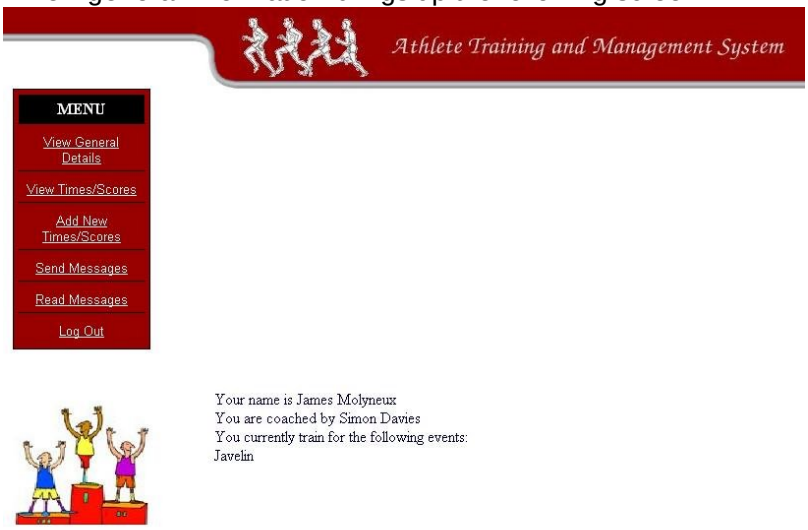
"Incorrect Password, Please Try Again!"

A successful login will bring you to the welcome page:

Using the System

You can navigate around the system using the menu on the left.

1. Clicking on “View general Information brings up the following screen:



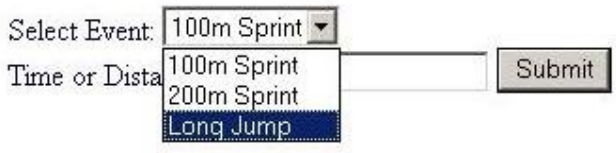
Here you can see who you are (who is logged on), who your coach is, and what events you compete in.

2. Clicking on “View Times/Scores brings up the following screen:

Here you can see the times you have entered into the system

3. Clicking on Add Times/Scores brings up the following screen

From here you can select one of your registered events from the drop down box, and enter a time to enter into the database, as shown below

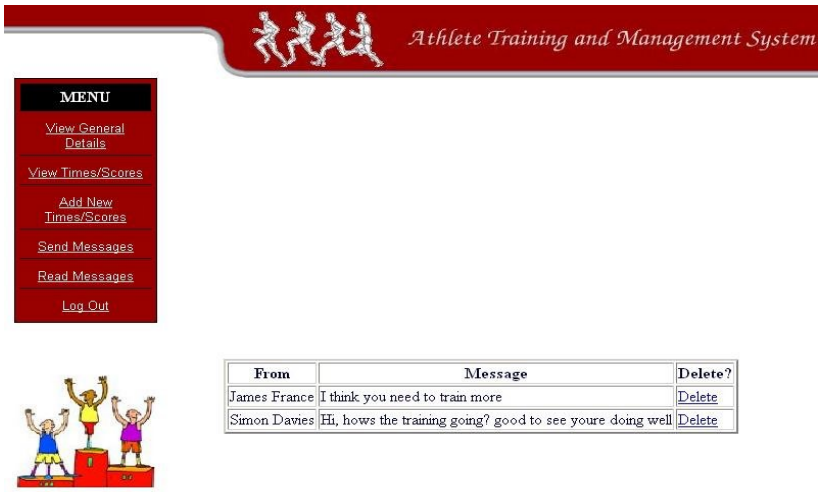


To add the result to the database, simply click submit.

4. Clicking on Send Message brings you to the following screen.

Simply enter the name of the coach you wish to send a message to, enter your message in the textbox, and click submit.

5. Clicking Read Message brings you to the following screen, which displays your messages.



Clicking on delete removes the message, and it is lost forever

6. Clicking on Logout removes the cookie and exits you from the system.

Coaches

Logging In

Coaches enter the system at the index page:

Athlete Training and Management System - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://wwwdev.comp.leeds.ac.uk/scs1jdm/
Athlete Training and Management System
MENU
Athlete Log In
Coach Log in
Welcome to the Athlete Training and Management System.
Please Log in using the menu on the left

And navigate to their login page by clicking the appropriate link in the menu on the left

Enter the relevant information in the boxes provided. Should you enter your username incorrectly you will see the message

"Coach Not Found, Please Try Again"

Should you enter an incorrect password, you will see the message

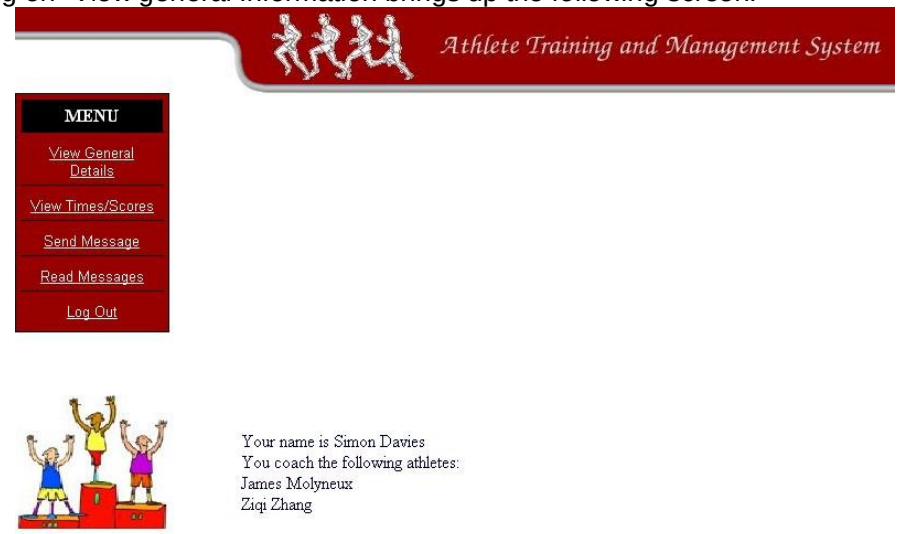
"Incorrect Password, Please Try Again!"

A successful login will bring you to the welcome page:

Using the System

You can navigate around the system using the menu on the left.

1. Clicking on “View general Information brings up the following screen:



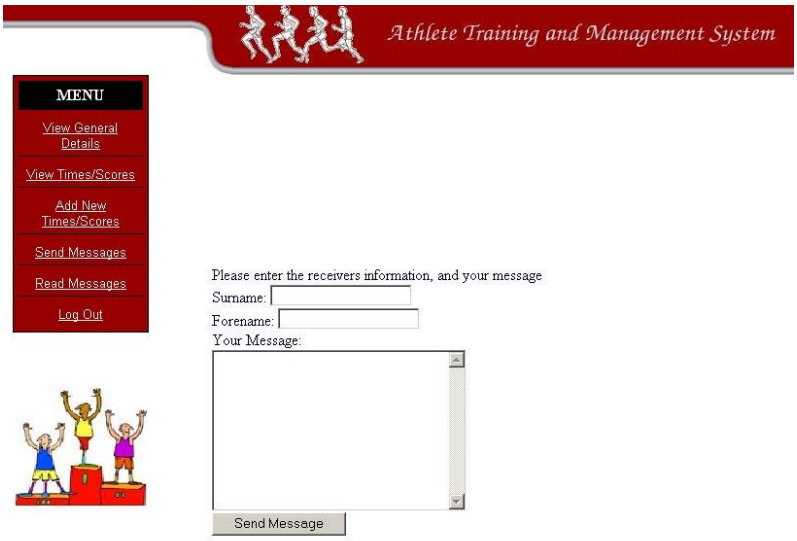
Here you can see who you are (who is logged on), and who you coach

2. Clicking on “View Times/Scores brings up the following screen:

Enter the name of the athlete whose times you wish to see. Assuming you are this athletes coach, their scores will be displayed as shown below. If you are not the coach of the athlete, you will see the error message

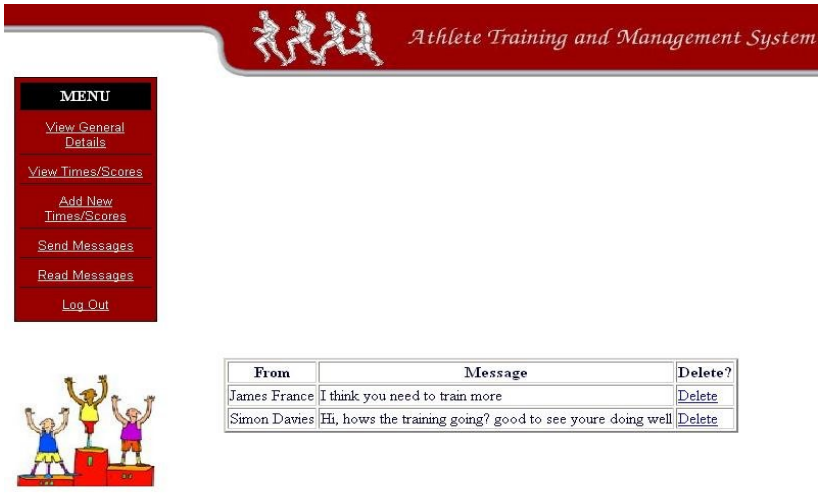
"Sorry, but you don't coach this athlete"

4. Clicking on Send Message brings you to the following screen.



Simply enter the name of the athlete you wish to send a message to, enter your message in the textbox, and click submit.

5. Clicking Read Message brings you to the following screen, which displays your messages.



Clicking on delete removes the message, and it is lost forever

6. Clicking on Logout removes the cookie and exits you from the system.